Mapping Web Personal Learning Environments

Matthias Palmér, Stéphane Sire, Evgeny Bogdanov, Denis Gillet and Fridolin Wild

[1] Royal Institute of Technology (KTH), Stockholm, Sweden,
matthias@nada.kth.se
[2] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland,
{stephane.sire, evgeny.bogdanov, denis.gillet}@epfl.ch
[3] The Open University, Milton Keynes, United Kingdom,
f.wild@open.ac.uk

Abstract. A recent trend in education and in web development is to build learning environment on top of web platforms which are carefully designed to host a plurality of software components (sometimes called widgets or plugins) which can be organized or combined (mashed-up) at user's convenience to create personalized environments. The degree of personalization can depend on the role of the users such as in traditional virtual learning environment, where the components are chosen by a teacher in the context of a course. Or, it can be more opened as in a so-called personalized learning environment (PLE), for instance as in the case of a personalized homepage hosed on a portal such as Netvibes. It now exists a wide array of available web platforms exhibiting different functionalities but all built on the same concept of aggregating components together to support different tasks and scenarios. This article shows that recent developments and standardization efforts allow to map the functionality of these platforms onto more or less independent dimensions which are becoming the building blocks for constructing a personal learning environment.

Keywords: widget, mashup, web, comparison, PLE.

1 Introduction

One of the key outcomes of learning is the construction of a learning environment, i.e. that set of tools that brings together people and content artefacts in learning activities to support them in constructing and processing information and knowledge. These environments are distributed and networked by nature. When looking at personal arrangements in this learning ecosystem, i.e. an individual's selection of tightly- and loosely-coupled tools, close and distant contacts, both created and consumed objects, used for and in main as well as side activities, we speak of a personal learning environment (Wild, 2008).

Looking at the digital parts of this environment, a rich set of implementation approaches can be found in previous work on personal learning environments. Early work (e.g. Liber, 2000; Kearney et al., 2005) focuses mainly on conceptual issues, the

next phase is characterised by an emphasis on interoperability issues (Downes, 2005; Wilson, 2005; Wilson et al., 2007) and a stronger emphasis of linking personal learning environments to social software: data interoperability, most notably RSS/ATOM-based aggregation, and service integration of web-services such as the storage/retrieval services offered by the Flickr API.

Whereas in recent advances Wilson et al. (2007) propose to differentiate implementation strategies into coordinated use, simple connectors for data exchange and service interoperability, and abstracted and generalized connectors in form of conduits, we propose within this contribution to further differentiate the latter two into a set of six dimensions with corresponding implementation features. These six PLE dimensions encompass screen, data, temporal, social, activity, and run-time. At the end of the paper we use these dimensions and features to compare six different platforms. We believe the result can be used both to make decisions on which platform to use today as well as se trends and identify areas where further investigation are needed to pave the way for better PLEs.

In our subsequent elaboration on these dimensions of a web PLE architecture, we will be using the following specific terminology. First of all, the software components sometimes referred to as applications, plugins or widgets that are hosted in the PLE will be refered to as simply widgets. Second, we will refer to the server side code that manages the main functionalities of the PLE as the PLE engine. Third, the settings used to initialize and deploy the hosted widgets will be referred to as widget preferences and taken together as the PLE configuration which might be stored locally in the PLE engine or remotely via a dedicated configuration service. Fourth, the web container which renders and executes the widgets on the client-side together with common facilities such as navigation between the widgets will be referred to as the PLE container. Finally, PLEs may provide a way to organize sets of widgets together, for instance into tabs, which can constitute a learning context, sometimes with support for collaboration. We will refer to such sets of widgets as PLE spaces.

2 Overview of Dimensions

The Table 1 summarizes six dimensions of functionality to be used when measuring the PLE characteristics of web platforms. The definitions of the dimensions are made to both capture as many relevant features as possible as well as to make them independent, implying that web platforms can support any combination of them.

These dimensions are most of the time independent and they are not all necessary to build a PLE. For instance a user that would select a Netvibes personalized homepage as their PLE would more or less only use the screen dimension. It is even conceivable that somebody uses a PLE with none of these dimensions. This is the case for a user that select a simple blogging tool such as Wordpress to self-reflect on her learning process by writing text snippets (without plugins and comments otherwise there would be some elements of the screen and social dimensions).

Table 1. The dimensions for building web Personal Learning Environments.

Dimension	Definition	Potential Standards
Screen	Organization of several widgets within a PLE in a spatial manner.	W3C Widgets 1.0 Google Gadget API Google GadgetTabML Netvibes UWA OpenAjax Metadata 1.0
Data	Interoperability of data and metadata across widgets and underlying services Includes issues with cut & paste, drag & drop, data formats, protocols, semantics.	Various Data and metadata standards such as RSS, SCORM, Dublin Core RDF, HTML5 Dn'D OpenAjax Hub 2.0 Google gadget pubsub
Temporal	Updates to widget configuration, state or data is more or less synchronous with other active users who share the widget instance.	COMET/Reverse Ajax XMPP, XMPP over BOSH Google Wave Federation Protocol
Social	Interoperability of user identity, profile information and list of friends. Ability to define some group contexts for sharing widget events, state or data.	OpenID (portable profiles) OpenSocial API Portable Contacts Facebook Connect (Friend Linking and Social Distribution) FOAF
Activity	The applications in use in the PLE can be controlled through scripts that engage the user into learning activities.	WS-BPEL (business oriented) IMS Learning Design Specification (targeted at VLE and design-time oriented, not run-time)
Runtime	Cross PLE interoperability allows you to exchange one rendering and execution platform or its parts with another.	W3C Widgets 1.0: Packaging and Configuration OPML Open Ajax Mashup Reference Application

However, we believe that the power of web PLEs will be to support several of these dimensions. This will allow more powerful widgets, such as for instance collaborative widgets by including elements of the screen, temporal and social dimensions. But it will also make the web PLEs more reliable environments, for instance the data dimension will make your data more portable to avoid data lock in. Furthermore, the runtime dimension will allow you to switch to another PLE of your choice with minimal migration issues and the feature of being able to collaborate

across different PLEs will allow you to stay even when your collaborators are using different PLEs.

The diagram below shows these dimensions graphically in what could be an abstract view of a generic PLE. In this diagram we have made explicit that we see the PLE as an Integrated Development Environment (or IDE), this is because it allows you to develop a learning environment to fit your needs rather than force you to be satisfied by what is given. For instance in a PLE with social integration, a part of the user interface is dedicated to invite friends and to accept invitations. Similarly in a web personal Home page a part of the user interface is dedicated to browse widgets and to place new widgets on a grid on the screen.

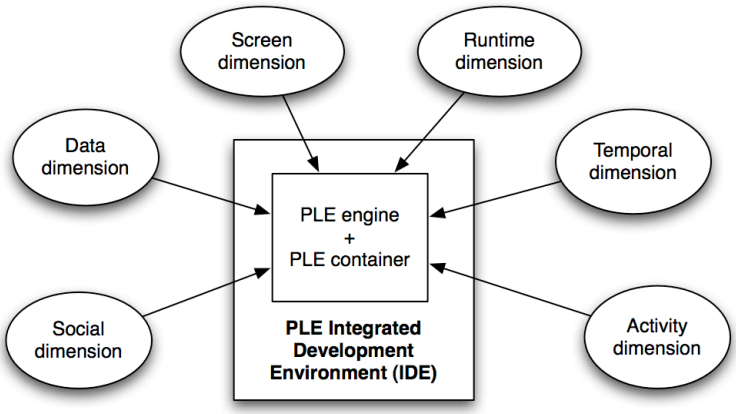

Fig 1. The dimensions for building web Personal Learning Environments.

In the next sections, we examine each dimension in greater detail.

3 Screen Dimension

Many platforms allow multiple distinct software components to exist side by side within a container application. Examples range from simple pasting of html-snippets inside blog posts to advanced standard based widget containers or even web Desktops that comes with a full set of tools. Below we outline a few possible features of a container application with respect to the screen dimension.

Shared screen. Many Web sites, such as video sharing sites, allow their users to cut and paste short snippets of (X)HTML code that allow to project their content. These snippets typically uses either an <iframe> tag to embed a separate web page / application, <object> tag for embedding of flash or java applets, a <script> tag for loading a separate javascript that dynamically builds up the user interface, or any combination of the above. For blogging or wiki platforms (e.g. WordPress, MediaWiki) this approach is a good alternative. For Content and Learning

Management Systems it is more common with plugin based approaches. In general, plugins requires more administration or even programming skills to be made available. Depending on the level of automation of the system you might need access to the underlying installation to reconfigure or activate new plugins.

Widget standards. The most limiting factor of plugins are that they are by definition limited to a single platform. If a plugin infrastructure grows beyond the boundaries of a single platform and brings with it a specification and good documentation it is perhaps more reasonable to talk of a de facto standard, lets call it a widget standard in this case.

Google gadgets are a typical example of a de facto standard while for example Netvibes UWA, Wordpress plugins and Facebook applications are somewhere in between. In addition there are standardization initiatives like the W3C widget 1.0 specification (W3C, 2009b), the OpenAjax metadata specification (OpenAjax Alliance, 2009b) or the BONDI specification (Open Mobile Terminal Platform forum, 2009). The benefit of adopting widget standards are that they help to establish an ecosystem of widgets that can run on many platforms, for example Google gadgets can run both in Wordpress, Netvibes and Facebook.

Layout of widgets. Platforms that have a clear focus on a single activity, like writing blog posts or editing wiki pages will probably not have more than a few widgets active at the same time. However, with more advanced PLEs it must be possible to focus on different kind of activities which implies a wider range of supportive widgets being accessible. Hence, the need for choosing your own layout and order the widgets according to your own needs increases. A common approach is to allow widgets to be placed on multiple dashboards that are available as tabs.

Web Desktop. To lower the barriers for non advanced users a truly capable PLE probably need to provide a default set of generic tools similar to a regular pc desktop. Even though these tools are widgets as well they are likely to be available from a menu, to minimize clutter, rather than laid out on a gigantic dashboard by default. This brings us close to the concept of Web Desktop or Web Operating System which are web applications that mimics the desktop, examples include EyeOS and G.ho.st. There are several aspects of the regular Desktop experience that could be brought along, for example the idea of sessions where widgets can be opened or closed and access to content (independent of widget) via something like a file manager.

4 Data Dimension

There are many forms of data portability which have been described elsewhere (Turnitsa, 2005). In web 2.0 data portability is often seen as data created or stored within application A can be read and/or copied/moved into application B and interpreted by application B, see for example the vision statement from the

DataPortability project. What we see today is mostly simple widgets using established standards such as RSS while more advanced widgets introduce their own the choice of data formats and services for content and own vocabularies for preferences. The latter choices are nearly always made based on the perspective of a individual widgets and not a wider widget landscape. The result are widgets that have little or no knowledge of other widgets or even the surrounding context and the data is often specific for a single widget. More recently we have seen the emergence of client-side communication protocols that allow applications integrated at the presentation level (screen dimension) to exchange data on user's behalf (autonomously or with methods such as Dn'D).

A web PLE could provide good supporting structures that both enable widget developers to provide more cross widget features as well as help them to make good decisions on data portability. Below we identify four such supporting structures, features, that a web PLE could provide with respect to the data dimension.

Inter-widgets communication. Some recent containers have started to propose client-side data portability, that is the ability for applications to exchange data directly within a host container without interfering with their server-side implementation, see (Sire & al., 2009). Such features can be supported with the integration of a communication layer within the host container, examples include Google gadget "pubsub" (Google, 2009a) and the OpenAjax Hub Publish/Subscribe APIs (OpenAjax Alliance, 2009a).

Drag and drop. With the upcoming HTML5 (W3C, 2009a), drag and drop will be available natively in the browser. But it is also possible to achieve similar effects already today via a Javascript communication layer in the client.

PLE data manager. An increasingly common approach to data integration consists in providing storage services apart from the applications that need storage. The storage services allow several applications hosted on a container, or several instances of the same application hosted on the same engine, to share a common data store. This service can be part of the PLE, such as in OpenSocial containers with the data persistence API, or they can be offered by independent providers such as with the Google Wave Federation Architecture. A PLE data manager could also start by providing support for a few common protocols such as Atom Publishing Protocol and search APIs (e.g. OpenSearch, SQI) to allow the development of unified search services into a PLE.

Linked data support. The main difficulty of client-side data portability is to determine the compatibility of two components before establishing a communication channel between them. Typically they require the agreement on common microformats to exchange data (e.g. hCalendar, FOAF) or meta-data about the data (e.g. SCORM, Dublin Core). A semantic driven approach using basic HTTP with common RDF properties (linked data) may offer greater flexibility and scalability as

well as enable both interoperability on the fly and automatic data mash-ups. As the world will continue to contain non-semantic data sources in the forseeable future a semantic gateway will be needed. SA-REST (Lathem & al., 2007), GRDDL, RDFa and similar initiatives could be really useful for the realization of such a gateway.

5 Temporal Dimension

Collaborative aspect of web PLEs brought a new dimension to people. It is not enough to only assign users the specific rights over a particular shared object, but it is equally important to notify users about current content and changes in this object. In other words, the state changes in a shared object should be propagated to all people engaged.

The initial architecture of the Internet allowed update to object state to happen only on page reload. Thus, if two people worked on the same document, the first person could see the changes of the second person only after clicking the refresh button in the browser. Initial workarounds such as polling was unsophisticated but recently, the situation has dramatically improved. With the advent of online chat applications, new protocols were invented (COMET, Reverse Ajax, XMPP, etc.) which allowed updates to be synchronous propagated. With these new techniques in mind we outline four features that a PLE can support regarding the temporal dimension.

Push data updates. A common approach to synchronous updates is to have a Javascript API that allows widgets to push data to other instances of the same widget within a well defined context. All users that have that widget in the same context will see the update without reload. The Wookie engine provides exactly this functionality as an extension to the Widget 1.0 APIs and Events specification. The Google realtime gadgets API is another initiative that today allows widgets instances running inside a Google online chat to exchange data although it has been stated that it will soon work on both the iGoogle and Orkut platforms (Google, 2009b).

Push preference updates. Many simple widgets have no data (except static resources) except preferences. A typical example is a map widget that shows a specific location. A user who wants to show a specific position in the map widget should be able to do so without forcing all other users sharing the context to reload their PLEs. A more advanced example could be that the PLE space must change to reflect a new need, for example add and configure a new widget.

Real time data updates. The distinction between push and real time data updates are important for two reasons. First it puts higher demands on the infrastructure since delays are more clearly noticed. Second depending on the character of the data, data updates might need to be merged and locking mechanisms to avoid conflicts is not an option due to the real time aspect. For example, if two people edit in the same place in a collaborative text document there is a risk for conflicts but if you have a voting

application the votes should be stored separately, making conflicts impossible. Google Wave is an example of a platform that provides real time updates.

Data and preferences history. With data updates coming from different users the need to be able to go back to view or even revert to an older version of the history becomes more important. Optimally the history should include both the widget preferences, the wider PLE configuration as well as other data. Examples of systems that include history today are wikis, content management systems, and Google Wave via it's playback mechanism. Google Wave playback is especially interesting since the versions are not necessarily distinct in the traditional document centric perspective, instead there is a continuum of real time updates that can be scrolled through.

6 Social Dimension

Web 2.0 applications explicitly model the concept of the user and of her list of friends, or followers. This information is used to exploit the social graph of the user when sharing application data and posting notifications. Generally speaking, the social graph information is used by the social container to dynamically define different groups when a user is interacting with an embedded application, either on his own page, most of the time called the user's profile page, or on the pages of other users. We have identified four features that a web PLE can support in the social dimension.

List of Friends. This is the most basic level where a system supports users to have a list of friends. Many systems requires a step of confirmation before friendship is established and sometimes the friendship is qualified into more specific relation categories such as friends, relatives, colleagues, etc. To avoid the hassle of recreating your list of friends manually, many systems provide import/export mechanisms supporting formats such as vCard or FOAF.

Friends server. Unless users are very organized, they often run into synchronization issues when importing/exporting their list of friends between systems. Another approach, more modern and effective, is to have his list of friends in a single system and make them appear in other systems via a protocol such as Facebook connect or OpenSocial.

Access control. The OpenSocial protocol defines several groups such as owner, viewer, friends of owner, friends of viewer (OpenSocial Foundation, 2009b). These groups are used to define access control policies to application data, and to define data storage mechanisms that allow users to send data to their friends, or to consult data from their friends, applying complex rules for developers. For instance a common rule to facilitate the viral spreading of applications within social networks is that it is usually required to have installed the application on her own profile before being able

to post data through this application on a friend's profile. Some rules are also used to govern reading access to the activity streams of a user and to manage writing of notifications from a user into other user's news feeds (Facebook Developers, 2009). The goal of these rules is to leverage the social graph of the user for making decision and proposing recommendations (OpenSocial Foundation, 2009a).

Independent groups. We can easily see how the social graph dimension could also leverage the educative power of a PLE by allowing to support learning activities in groups. However our impression is that currently the flexibility to create and manage user's profile which has been gained with protocols such as OpenId is still missing for managing group's profile in a way that is independent of the PLE engine. More flexibility will be needed to define different groups which may be related to different user's activities and to be able to import these groups into different PLE engines depending on the context. One approach to solve this could be to do an extension to OpenId, perhaps called GroupId.

7 Activity Dimension

Activity theory states that an activity is shaped by its surroundings (Leont'ev, 1947; Engeström, 1987). For example, tools do have certain affordances: a door knob lends itself to opening. On the contrary, activities do also shape their surroundings: they can result in the construction of a tool. Below we identify four features that provides various levels of support for activities in the surrounding Web PLE.

Manual guide. The simplest approach for guiding the user through a series of actions is simply to have instructions easily available in the environment. Simple checkboxes or text areas where the user can mark and reflect on progress are a nice addition. If there are decision points a more complex view is needed where branches can be opened or closed, again based on manual input. Even though a manual guide is achievable in a regular widget, giving it a more prominent position in the environment might enhance the feeling of guidance for the user.

Flow enables widgets. Based upon a script and manual input of the user, certain widgets can become enabled or visible, leading the way for the user. Some care has to be taken since when enabling web applications involved in learning activities to become part of scripted activities (Dillenbourg & Jerman, 2007), matters of orchestration of how users should use a set of tools and services to achieve a certain learning goal demand attention.

Scripted inter-widgets data flow. In addition to simply activating or enabling widgets based on manual input you can let scripts do it for you, based on the resulting produced/transformed/aggregated data from other activities. Furthermore, the widgets preferences or even starting data can be initialized from these earlier activities. The

difference to the inter-widgets communication feature in the data dimension is that the data flow is driven by the script, not the individual widgets. The script might even take into account the social dimension, making use of different users contribution. SCORM runtime and Learning Design are two example of standards that try to encode the scripting logics in combination with content rather than widgets. Mupple (Wild et al., 200) is one example of a activity scripting environment in a widget context. The underlying learner interaction scripting language (LISL) allows to specify action/outcome/tool bindings which can render the user interfaces of web applications into one common dashboard like environment thereby providing a kind of human self-instructing and scaffolding into defining intended outcomes.

Recommendations. A more advanced case is when the decisions, data flow and recommendations are performed by a independent software agent. It might, for instance, consult remote services, investigate user history, collect attention metadata, know about desired outcomes and aspired goals, and then make inferences to provide guidance in the form of recommendations. There is many things that can be done on this level, for instance (Kildare et al., 2009) deals with explicating interaction norms and rules that can thereby be monitored automatically, thus relieving the user of certain regulation work within collaborative scenarios. In this case, rules are formulated in statements for the Java Expert System Shell (JESS).

8 Runtime Dimension

In the future we do not expect that there will be a single web PLE that everyone uses. Instead, we find it more realistic that there will be many distinct web PLEs that compete with slightly different functionality, design and interaction paradigms. Furthermore, we believe that the full strength of having a range of alternative web PLEs is not reached until it is both easy to move between PLEs and also to collaborate across PLE boundaries. Thus, we believe it is crucial that a widget or an entire PLE space in one PLE can be easily experienced in another PLE. Below we list four features that a PLE can support with respect to the runtime dimension.

Feed export and import. Feeds are today the bread and butter for most widgets, and many PLE platforms provide specific support, for instance, allowing you to add a feed directly rather than adding and configuring a widget to display it. It is also common to provide import and export of OPML files (listing feeds with feed names, possibly in groups) allowing simple migration to other PLE environments or dedicated feed readers.

Generic export and import. Even though feeds are important there are many widgets that use other forms of data that will not survive the transfer via an OPML file. Providing a lossless export and import of a PLE configuration from the same system is the first step, iGoogle provides GadgetTabML that does this. As the second step,

the format needs to be widely accepted, hopefully even standardized, to allow lossless (or close to it) export and import between PLEs. Although, even if such a lossless export and import is supported between PLEs, it depends on both widget standards as well as authentication and authorization protocols to be in place already.

External configuration. By keeping the PLE configuration in a separate service distinct from the PLE engine we accomplish three things. First, users need not import or export, they will have a single reliable access point to point their PLE to. Hence they could easily switch between or even use multiple PLEs in parallel to get the best out of each service. Second, following each others PLE spaces, or even collaborating, will be possible across PLEs. Typically a teacher could invite students into courses that they could partake in a PLE of their own choice. Third, keeping PLE configurations in a dedicated service will encourage additional levels of functionality such as a PLE configuration ecosystem where the use of templates could help users to get started and teachers to establish good practices.

Embedding. In the screen dimension section we discussed different alternatives to how software components, such as widgets, can appear together in a PLE, but we did not discuss how an entire PLE could be embedded into another environment. The approach to paste HTML snippets that via iframe, embed or script tags is viable here as well. A simple example that people are using is running Facebook as a tab in iGoogle via the canvas view (a full screen iframe widget). Clearly iframe embedding is rather weak from an integration perspective. When the host environment is a PLE it would be beneficial to have a deeper integration, especially along the social and data dimensions outlined above. Such a deeper integration would rely on Javascript APIs and Javascript wrappers probably in combination with a few standardized protocols. One possible example of this is the Google Wave embed API, however it is still unclear if Google Waves should be compared with an entire PLE or a very advanced widget. The Open Ajax Mashup Reference Application proposes a portability solution to individual widgets, as does the Widget Server in the Wookie widget engine. This could be generalized beyond single widget rendering to support full PLE configuration rendering.

9 Mapping Existing Web PLEs

In this section we will map six different platforms with respect to the six PLE dimensions. The platforms we map are the personal/social start pages iGoogle (IG) and Netvibes (NV), the Learning Management System Moodle combined with the Wookie engine (M&W), the collaboration platform Google Wave (GW), the mashup platform Afrous (AF), and the web desktop G.ho.st (GH).

We have investigated the support for each of the four features per dimension (introduced above) and simplified the findings to whether it is supported or not. The investigation has been done by reading available documentation and testing the

platforms (except Google Wave as it is not available for the general public yet). As the features are not absolute there are situations when there is only partial support (reported as supported) or that the feature is not applicable (then reported as not supported). A full explanation for why we have deemed that a platform has support or not for a specific feature is unfortunately out of scope of this paper although some cases are more or less explicit from the discussion in the sections above.

Table 2. The 6 mapped platforms.

Abr.	**Platform**	**Type**	**URL**
IG	iGoogle	start page	www.igoogle.com
NV	Netvibes	start page	www.netvibes.com
M&W	Moodle + Wookie	LMS	getwookie.org/moodle/
GW	Google Wave	communication and collaboration	wave.google.com/help/wave/about.html
AF	Afrous	mashup	www.afrous.com
GH	G.ho.st	web desktop	g.ho.st

It is important to notice that the results should not be seen as a total evaluation of these platforms as they might have other features not evaluated here. For example, Afrous seems to be a very capable client side mashup platform and G.ho.st a good web desktop, however, this does not necessarily make them good web PLEs as defined by our dimensions.

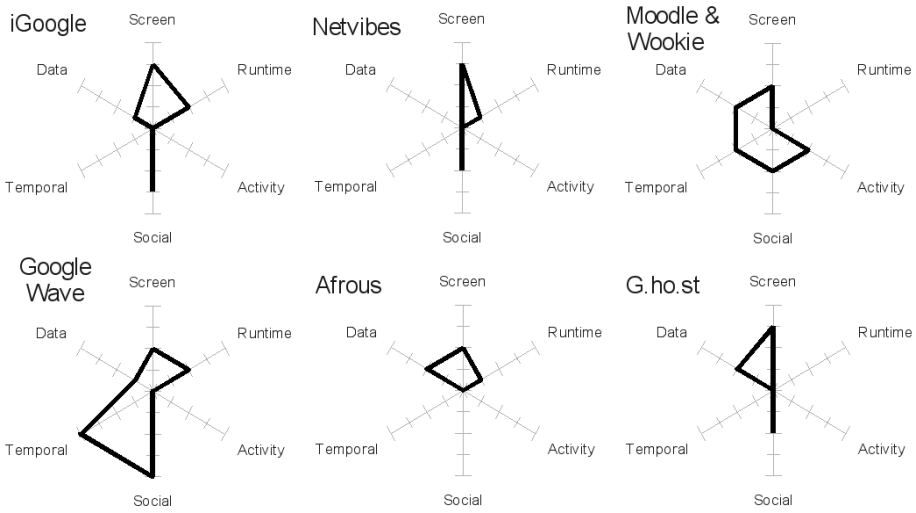

Fig 2. Comparing the features of the 6 platforms mapped onto 6 dimensions.

Table 3. Feature level details of the comparison of the 6 platforms.

Dimension	Feature	IG	NV	M&W	GW	AF	GH
Space	Shared screen	X	X	X	X	X	X
	Widget standard	X	X	X	X		
	Layout of widgets	X	X			X	X
	Web desktop						X
Data	Inter-widgets communication	X		X		X	
	Drag and drop			X	X		X
	PLE data manager					X	X
	Linked data support						
Temporal	Push data updates			X	X		
	Push preference updates			X	X		
	Real time data updates				X		
	Data and preference history				X		
Social	List of friends	X	X		X		X
	Friends server	X	X		X		
	Access control	X		X	X		X
	Independent groups			X	X		
Activity	Manual guide			X			
	Flow enables widgets			X			
	Scripted inter-widgets data-flow						
	Recommendations						
Runtime	Feed export and import	X	X			X	
	Generic export and import	X					
	External configuration				X		
	Embedding				X		

10 Conclusion

The Future of web PLE relies on the integration of several dimensions which have been historically developed separately for web applications such as widget portals, social networks and content management systems, and which are now converging. This allows new scenarios with more personalization and more group usage of web applications. However this approach will be successful and support the growth of web

PLE usage in education only if we can see a convergence between the multiple standardization efforts which are now underway at the World Wide Web consortium, Open Ajax alliance, Open Mobile Terminal Platform forum (OMTP), DataPortability project, at big companies such as Facebook and Google, the OpenSocial foundation, etc.

In this paper we have identified 24 features along six dimensions that can be used to map the functionality of web PLEs today. We have also investigated six platforms and presented the results as spider diagrams. If conclusions of the results are to be drawn, it can be noted that Google Wave and Moodle & Wookie are the only PLEs that are strong in the temporal, social and activity dimensions indicating a strong focus on collaboration. While the other platforms, iGoogle, Netvibes, Afrous, and G.ho.st have their focus on the data, screen, and runtime dimensions indicating a higher focus on personal customization. As we believe that good PLEs will need both strong personal customization and collaboration features, the conclusion is that there is still a lot of room for future web PLEs to realize that potential.

It should also be noted that Google Wave is at the time of writing a very young platform that we yet do not know very much about, it is very likely that it quite soon will score much higher on the data and runtime dimensions. However, that it will score higher in the screen dimension is not so obvious to us. If an individual wave is just a conversation or more what we have referred to as a PLE space, space remains to be seen.

The weakness of all but the Moodle & Wookie platform in the activity dimension may indicate that it should not be seen as a separate dimension of a PLE but as features of individual widgets. But, it could also mean that the platforms we have looked at are yet immature with respect to support for learning activities (which traditional Learning Management Systems excel at). The latter explanation is supported by experience from traditional Learning Management Systems where such features have only been realized with support from the surrounding platform and not being the sole responsibility of individual tools/widgets.

Finally, we hope that these features and dimensions could be further refined and perhaps be useful together with personal or organizational preferences for making decisions of which web PLEs to use. Perhaps can they also provide guidance in future research and development.

Acknowledgement

This work has been co-funded by the European Union under the Information and Communication Technologies (ICT) theme of the 7th Framework Programme.

References

Dillenbourg, P.; Jermann, P.: Designing integrative scripts. In F. Fischer, H. Mandl, J. Haake, and I. Kollar (Eds.): Scripting computer-supported collaborative learning: Cognitive, computational and educational perspectives, New York: Springer, pp. 277–302 (2007)

Downes, S.: E-learning 2.0. eLearn Magazine, 10, New York: ACM (2005)

Facebook Developers: Anatomy of an App (2009), online developers.facebook.com/get_started.php?tab=anatomy#news_feed

Engeström, Y.: Learning by expanding: an activity-theoretical approach to developmental research. Orienta-Konsultit Oy: Helsinki (1987)

Google: Gadget-to-Gadget Communication (2009a), online code.google.com/apis/gadgets/docs/pubsub.html

Google: Realtime Gadgets API (2009b), online code.google.com/apis/talk/gadgets_realtime.html

Kearney, D.; O'Hare, D.; McClure, G.; McKee, M.; Higgins, S.; Wishart, T.: Northern Ireland Integrated Managed Learning Environment (NIIMLE), Final Report of the NIIMLE project (2005), online www.elearning.ac.uk/mle/learner_recs/niimle/NIimle.pdf

Kildar, R.; Williams, R.N.; Hartnett, J.; Reimann, P.: Interaction Rules: their place in collaboration software, In: Mice, Minds and Society. The Computer Supported Collaborative Learning (CSCL) Conference 2007, 16-21 July, Brunswick, New Jersey, USA (2007)

Leont'ev, A.N.: Activity and Consciousness, Progress Publishers (1977)

Lathem, J., Gomadam, K. and Sheth, A.P.: SA-REST and (S)mashups : Adding Semantics to RESTful Services, In Proceedings of IEEE Int'l Conf. Conference on Semantic Computing, ICSC 2007, 17-19 Sept. pp. 469 - 476, IEEE (2007)

Liber, O.: Colloquia – a conversation manager, In: Campus-Wide Information Systems, 17(2), pp. 56 – 61, (2000)

Open Mobile Terminal Platform forum: BONDI Release 1.01 Specification (2009), online bondi.omtp.org/

OpenAjax Alliance: OpenAjax Hub Specification (2009a), online www.openajax.org/member/wiki/OpenAjax_Hub_Specification

OpenAjax Alliance: OpenAjax Metadata 1.0 Specification (2009b), online www.openajax.org/member/wiki/OpenAjax_Metadata_Specification

OpenSocial Foundation: Social Design Best Practices (2009a), online wiki.opensocial.org/index.php?title=Social_Design_Best_Practices

OpenSocial Foundation: The Persistence API (2009b), online wiki.opensocial.org/index.php?title=The_Persistence_API

Sire S., Paquier M., Vagner A. and Bogaerts J (2009).: A messaging API for inter-widgets communication. Proceedings of the 18th international conference on World Wide Web, Madrid, Spain, April 23, pp. 1115-1116, ACM New York, NY, US (2009)

Turnitsa, C.D.: Extending the Levels of Conceptual Interoperability Model. In Proceedings of IEEE Summer Computer Simulation Conference, IEEE CS Press (2005)

Wild, F.; Moedritscher, F.; Sigurdarson, S.: Designing for Change: Mash-Up Personal Learning Environments, eLearning Papers, Vol. 9 (2008)

Wilson, S.: Future VLE (2005), online zope.cetis.ac.uk/members/scott/blogview?entry=20050125170206

Wilson, S.; Liber, O.; Johnson, M.; Beauvoir, P.; Sharples, P.; Milligan, C.: Personal Learning Environments: Challenging the Dominant Design of Educational Systems, In: Journal of e-Learning and Knowledge Society, Vol. 2 (2007)

World Wide Web Consortium: HTML 5 A vocabulary and associated APIs for HTML and XHTML, Hickson I. and Hyatt D. (Eds.) (2009a), online www.w3.org/TR/html5/

World Wide Web Consortium: Widgets 1.0: Packaging and Configuration, Section 8.12 The feature Element, Cáceres M. (Ed.) (2009b), online www.w3.org/TR/widgets/#the-feature-element